

# HADOOP

## TP MapReduce

Nicolas LAGAILLARDIE

October 1, 2018

## 1 Objectif

Expérimenter le fonctionnement d'Hadoop à travers un exemple : calcul de la fréquence des mots présents dans les ouvrages de l'ABU puis calculer l'index des mots présents dans les ouvrages de l'ABU.

### 1.1 Définitions

**Hadoop** Un logiciel open-source pour réaliser des opérations informatiques des manière fiable, évolutif et distribuée.

**Pseudo-distributed mode** Un mode qui permet d'émuler le fonctionnement d'opérations distribuées sur plusieurs machines sur une même machine. On parle alors de pseudo-cluster.

## 2 Données expérimentales

**ABU** Un ensemble de 202 fichiers txt, provenant d'œuvres françaises. Les mots possèdent donc des accents et d'autres caractères, à prendre en considération dans le décompte.

## 3 Prérequis

L'ensemble des étapes sont détaillées sur le site internet d'Hadoop. Il s'agit d'abord d'avoir les logiciels prérequis : **ssh**, **rsync** et **Java**. Puis il faut assigner à Hadoop le chemin vers **Java**. En effet, Hadoop utilise Java pour fonctionner. Pour ma part, il a été nécessaire que j'utilise **JDK** et non **JSE**, car certaines classes manquaient à l'appel. Nous sommes ensuite prêts à lancer le **pseudo-cluster**.

## 4 Installation du pseudo-cluster

Comme nous n'avons pas immédiatement accès à un cluster complet, nous allons lancer un pseudo-cluster. C'est-à-dire que nous allons simuler un cluster, avec un *namenode* et un *datanode*.

Nous devons d'abord configurer le port d'utilisation d'Hadoop puis spécifier le nombre de *namenodes* utilisés. Dans notre cas, voici la configuration utilisée :

Modification du fichier *core-site.xml* :

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Modification du fichier *hdfs-site.xml* :

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

Il faut ensuite se connecter au *ssh* pour que Hadoop puisse communiquer avec ses autres pseudo *datanodes*.

```
$ ssh localhost
```

## 5 Exécution du *MapReduce*

Nous allons maintenant lancer notre script afin de pouvoir décompter le nombre de mots dans l'ensemble des fichiers txt. Il faut 6 étapes pour y arriver.

- a. Formatage de l'espace de stockage

```
$ bin/hdfs namenode -format
```

- b. Démarrage du *namenode* et du *datanode*

```
$ sbin/start-dfs.sh
```

- c. Création des dossiers *user* sur le cluster

```
$ bin/hdfs dfs -mkdir /user $ bin/hdfs dfs -mkdir /user/Lag
```

- d. Ajout des fichiers dans le cluster

```
$ bin/hdfs dfs -put ABU /user/Lag
```

| Permission | Owner | Group      | Size      | Last Modified | Replication | Block Size | Name              |
|------------|-------|------------|-----------|---------------|-------------|------------|-------------------|
| -rwxr-xr-x | lag   | supergroup | 46.6 KB   | Oct 01 21:52  | 1           | 128 MB     | adelaid2.txt      |
| -rwxr-xr-x | lag   | supergroup | 342.63 KB | Oct 01 21:52  | 1           | 128 MB     | aiglon1.txt       |
| -rwxr-xr-x | lag   | supergroup | 164.82 KB | Oct 01 21:52  | 1           | 128 MB     | amoursjaunes1.txt |
| -rwxr-xr-x | lag   | supergroup | 82.47 KB  | Oct 01 21:52  | 1           | 128 MB     | andromaque1.txt   |
| -rwxr-xr-x | lag   | supergroup | 131.03 KB | Oct 01 21:52  | 1           | 128 MB     | ane1.txt          |
| -rwxr-xr-x | lag   | supergroup | 299.7 KB  | Oct 01 21:52  | 1           | 128 MB     | annee1.txt        |
| -rwxr-xr-x | lag   | supergroup | 28.26 KB  | Oct 01 21:52  | 1           | 128 MB     | antiquites1.txt   |
| -rwxr-xr-x | lag   | supergroup | 654.8 KB  | Oct 01 21:52  | 1           | 128 MB     | archiduc1.txt     |
| -rwxr-xr-x | lag   | supergroup | 404.72 KB | Oct 01 21:52  | 1           | 128 MB     | arebours1.txt     |

Figure 1: Contenu du cluster

- e. Compiler puis exécuter notre script java

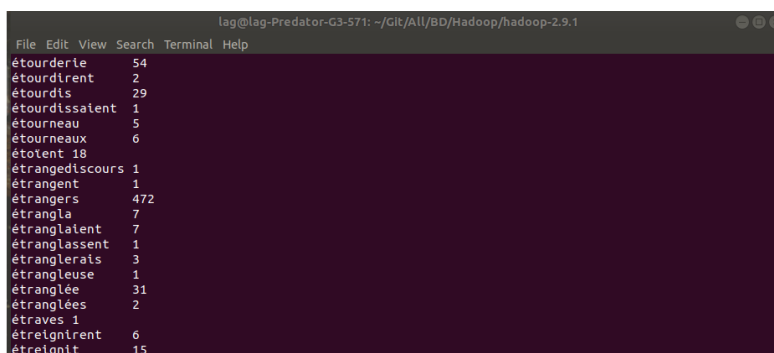
```
$ bin/hadoop com.sun.tools.javac.Main WordCount.java
$ jar cf WC.jar WordCount*.class
$ bin/hadoop jar WC.jar WordCount /user/Lag/ABU
/user/output
```

- f. Récupérer puis afficher les résultats

```
$ bin/hdfs dfs -ls /user/output
$ bin/hdfs dfs -cat '/user/output/part-*'
```

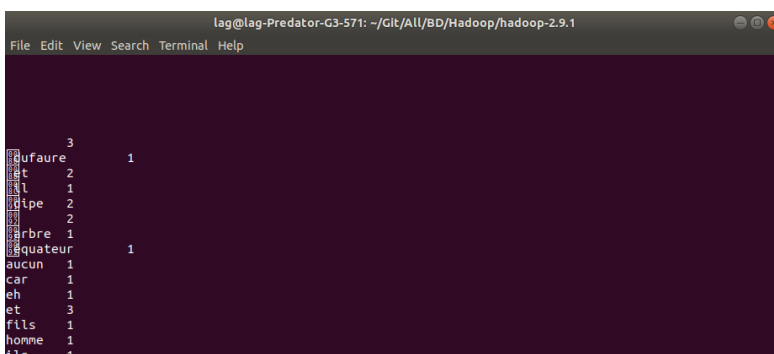
## 6 Premiers résultats

Suite au lancement de la première version du script, voici les résultats : ils sont globalement satisfaisants (voir Figure 2, mais de nombreux problèmes persistents (voir Figure 3).



|                 |     |
|-----------------|-----|
| étourderie      | 54  |
| étourdirent     | 2   |
| étourdis        | 29  |
| étourdissaient  | 1   |
| étourneau       | 5   |
| étourneaux      | 6   |
| étoient         | 18  |
| étrangediscours | 1   |
| étrangent       | 1   |
| étrangers       | 472 |
| étrangla        | 7   |
| étranglaient    | 7   |
| étranglassent   | 1   |
| étranglerais    | 3   |
| étrangleuse     | 1   |
| étranglée       | 31  |
| étranglées      | 2   |
| étraves         | 1   |
| étrégnirent     | 6   |
| étrégnit        | 15  |

Figure 2: premiers résultats



|          |   |
|----------|---|
| dufaure  | 3 |
| et       | 1 |
| il       | 2 |
| ipe      | 1 |
| ipe      | 2 |
| ipe      | 2 |
| ibre     | 1 |
| equateur | 1 |
| aucun    | 1 |
| car      | 1 |
| eh       | 1 |
| et       | 3 |
| fls      | 1 |
| homme    | 1 |
| fls      | 1 |

Figure 3: premières erreurs

Dans cette deuxième capture d'écran, nous remarquons que certains caractères accentués sont mal représentés. Tandis que d'autres caractères que nous n'avons pas dû prendre en compte dans notre script pour séparer les mots créent des erreurs non voulues.

## 7 Correction et second résultats

Les modifications possibles concernent la séparation et le comptage des mots des différents fichiers.

Voici une nouvelle séparation possible, qui ne dépend pas de caractères que nous établissons nous-même, mais de l'implémentation même de l'encodage. Ainsi, le code suivant :

```
StringTokenizer itr = new StringTokenizer(
    value.toString().toLowerCase(),
    " \\t\\n\\r\\f.,;:-'\"\\\"!/?/[\\]# =/+ '* $ 0123456789
");
while (itr.hasMoreTokens()) {
    word.set(itr.nextToken());
    context.write(word, one);
};
```

Est remplacé par celui-ci :

```
FileSplit fileSplit = (FileSplit)context.getInputSplit();
String filename = fileSplit.getPath().getName();
Path filePath = fileSplit.getPath();
String fileName = filePath.getName();

valueOutFilename = new Text(fileName);

for (String word : StringUtils.split(valueIn.toString())) {
    context.write(new Text(word), valueOutFilename);
}
```

Nous avons ainsi une séparation mieux définie et une concaténation de tous les résultats, au lieu de plusieurs fichiers résultats avec peu de cohérence entre chacun d'eux.

La deuxième partie a consisté à modifier le *Reducer* afin qu'il comptabilise les fichiers contenant les mots et non le nombre d'occurrences de ces derniers. Voici le code que nous avons modifié :

```

private IntWritable result = new IntWritable ();

int sum = 0;

for (IntWritable val : values) {
    sum += val.get();
}

result.set(sum);
context.write(key, result);

```

En ce code :

```

HashSet<String>fileNamesUnique = new HashSet<String>();

for (Text fileName: valuesInFileNames) {
    fileNamesUnique.add(fileName.toString());
}

String fileNamesOut = new String( StringUtils.join(fileNamesUnique, "
/ ") );

context.write(keyInWord, new Text(fileNamesOut));

```

Remarquons que j'ai choisi d'utiliser la fonction de hashage présente dans Java, afin d'accélérer le traitement des données. Nous pouvons observer les résultats de l'exécution dans la figure 4 où nous pouvons constater que pour chaque mot, les fichiers le contenant sont affichés à la suite.

[illegible]

Figure 4: Index des mots

## 8 Conclusion

Hadoop est un puissant système et ce TP a pu démontrer une des utilisations possibles. A l'aide d'un *simple* script Java, nous avons été en mesure de créer un index des mots présents dans 202 fichiers txt. Nous avons pu aussi utiliser un compteur de mots performant mais légèrement défaillant, que nous avons corrigé par la suite.

Par ailleurs, l'installation et l'utilisation d'Hadoop est bien documentée et assez simple en mode Pseudo-cluster.